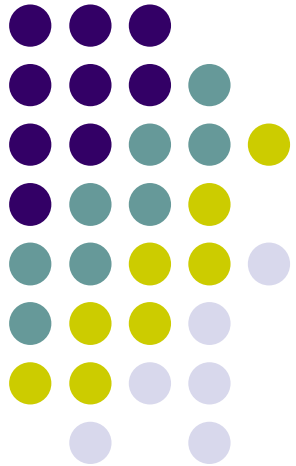
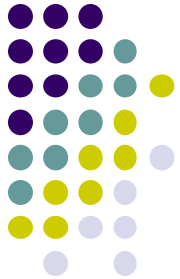
Basic Processing Unit

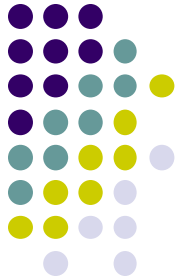


Fundamental Concepts



- Processor fetches one instruction at a time and perform the operation specified.
- Instructions are fetched from successive memory locations until a branch or a jump instruction is encountered.
- Processor keeps track of the address of the memory location containing the next instruction to be fetched using Program Counter (PC).
- Instruction Register (IR)

Executing an Instruction



- Fetch the contents of the memory location pointed to by the PC. The contents of this location are loaded into the IR (fetch phase).

$$IR \leftarrow [[PC]]$$

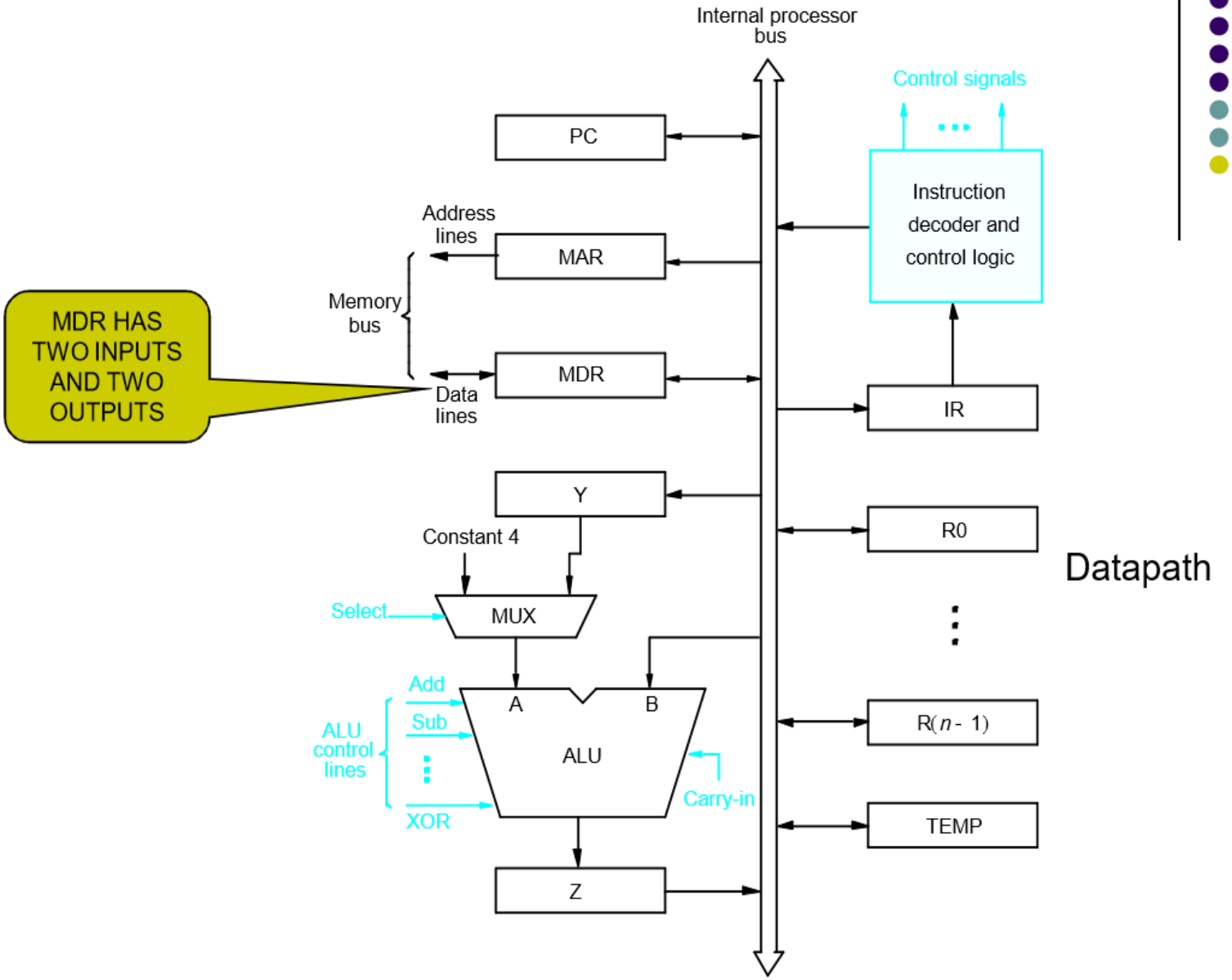
- Assuming that the memory is byte addressable, increment the contents of the PC by 4 (fetch phase).

$$PC \leftarrow [PC] + 4$$

(where each instruction comprises 4 bytes)

- Carry out the actions specified by the instruction in the IR (execution phase).

Processor Organization



Datapath

Registers:

- The processor registers R0 to Rn-1 vary considerably from one processor to another.
- Registers are provided for general purpose used by programmer.
- Registers Y, Z & TEMP are temporary registers used by processor during the execution of some instruction.

Multiplexer:

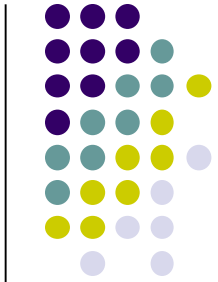
- Select either the output of the register Y or a constant value 4 to be provided as input A of the ALU.
- Constant 4 is used by the processor to increment the contents of PC.

ALU:

Used to perform arithmetic and logical operation.

Data Path:

The registers, ALU and interconnecting bus are collectively referred to as the data path.



Register Transfers

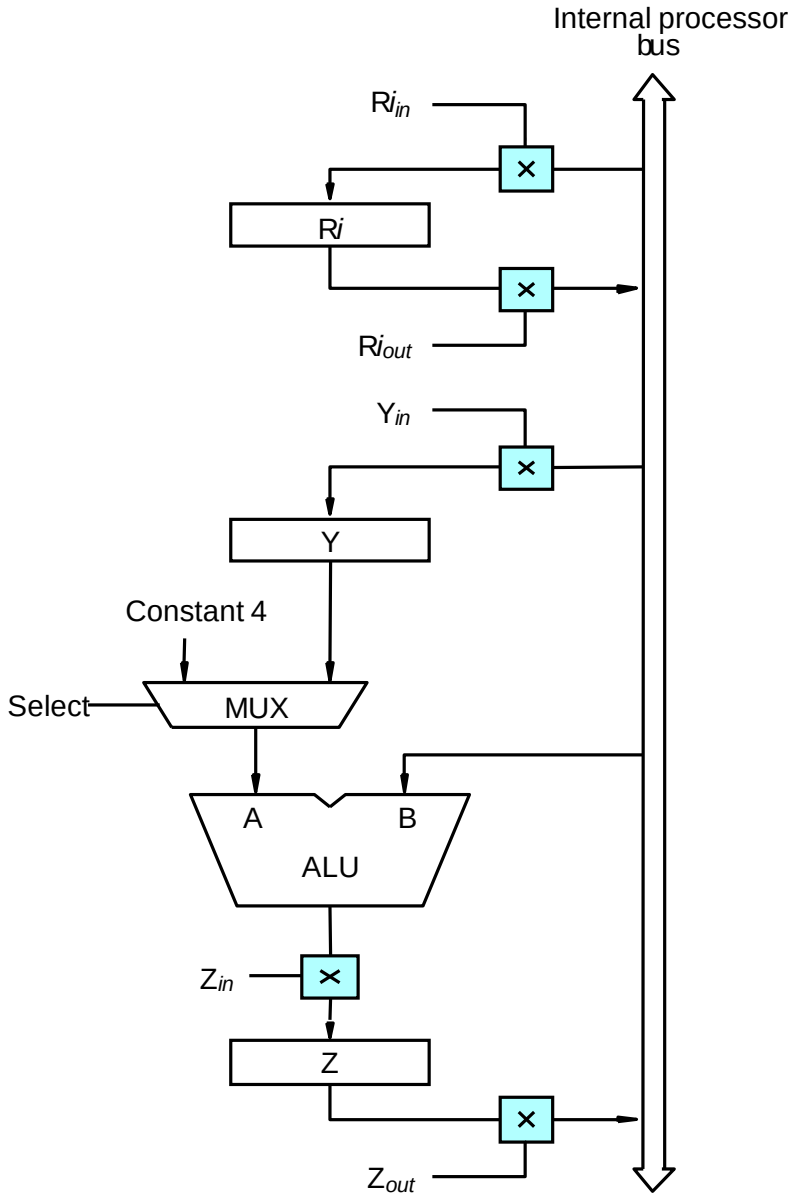
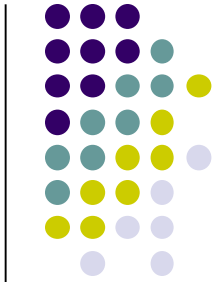


Figure 7.2. Input and output gating for the registers in Figure 7.1.

- The input and output gates for register Ri are controlled by signals is R_{in} and R_{iout} .
- If R_{in} is set to1 - data available on common bus are loaded into Ri.
- If R_{iout} is set to1 - the contents of register are placed on the bus.
- R_{iout} is set to 0 – the bus can be used for transferring data from other registers .

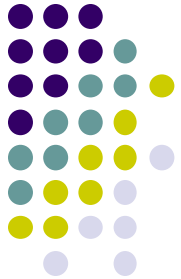
Data transfer between two registers:



Transfer the contents of R1 to R4.

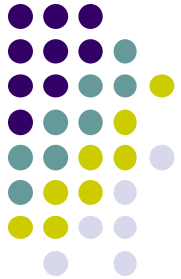
1. Enable output of register R1 by setting $R1_{out}$=1. This places the contents of R1 on the processor bus.
2. Enable input of register R4 by setting $R4_{in}$=1. This loads the data from the processor bus into register R4.

Performing an Arithmetic or Logic Operation



- The ALU is a combinational circuit that has no internal storage.
- ALU gets the two operands from MUX and bus. The result is temporarily stored in register Z.
- What is the sequence of operations to add the contents of register R1 to those of R2 and store the result in R3?
  1. $R1_{out}$, Y_{in}
  2. $R2_{out}$, Select Y, Add, Z_{in}
  3. Z_{out}, $R3_{in}$

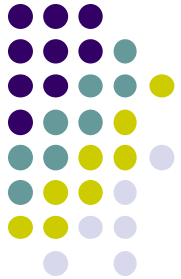
Fetching a Word from Memory



- Address into MAR; issue Read operation; data into MDR.

Figure 7.4. Connection and control signals for register MDR.

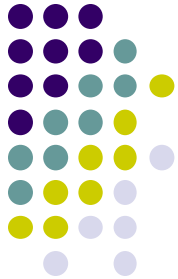
Fetching a Word from Memory



- The response time of each memory access varies (cache miss, memory-mapped I/O,…).
- To accommodate this, the processor waits until it receives an indication that the requested operation has been completed (Memory-Function-Completed, MFC).

- **Example: Move (R1), R2**
  - MAR ← [R1]
  - Start a Read operation on the memory bus
  - Wait for the MFC response from the memory
  - Load MDR from the memory bus
  - R2 ← [MDR]

Fetching a Word from Memory



Example: Move (R1), R2

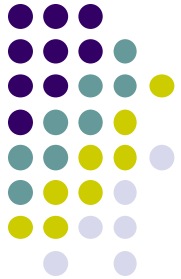
- MAR ← [R1]
- Start a Read operation on the memory bus
- Wait for the MFC response from the memory
- Load MDR from the memory bus
- R2 ← [MDR]

Assume MAR is always available on the address lines of the memory bus.

- Move (R1), R2

1. $R1_{out}$, MAR_{in}, Read
2. MDR_{inE}, WMFC
3. MDR_{out}, $R2_{in}$

Storing a word in memory



- Address is loaded into MAR
- Data to be written loaded into MDR.
- Write command is issued.

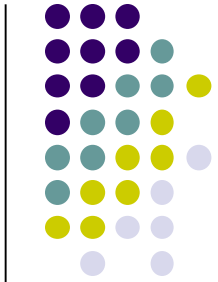
- **Example: Move R2,(R1)**

  $R1_{out}$,MAR_{in}

  $R2_{out}$,MDR_{in},Write

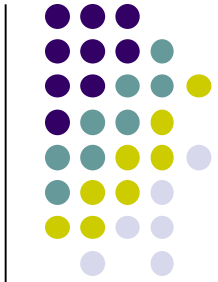
  MDR_{outE}, WMFC

Execution of a Complete Instruction



- **Example: Add (R3), R1**
- Fetch the instruction
- Fetch the first operand (the contents of the memory location pointed to by R3)
- Perform the addition
- Load the result into R1

Execution of a Complete Instruction



Add (R3), R1

Step	Action
1	PC_{out} , MAR_{in} , Read, Select4, Add, Z_{in}
2	Z_{out} , PC_{in} , Y_{in} , WMF C
3	MDR_{out} , IR_{in}
4	$R3_{out}$, MAR_{in} , Read
5	$R1_{out}$, Y_{in} , WMF C
6	MDR_{out} , SelectY, Add, Z_{in}
7	Z_{out} , $R1_{in}$, End

Question

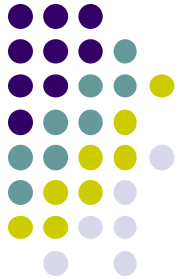
- Write the control sequence to execute

1. Add the (immediate) number NUM to register R1.

2. Add the contents of memory location NUM to register R1.

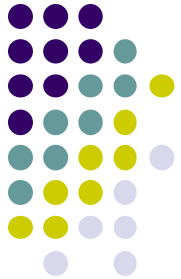
3. Add the contents of memory location whose address is at memory location NUM to register R1.

Solution 1



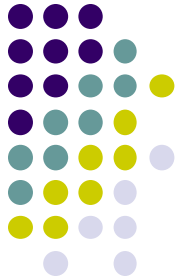
- PC out, MAR in, Read, Select4, Add , Zin
- Zout, PCin, Yin, WMFC
- MDRout, IR in,
- IR NUMout , Yin
- R1 out, Select Y, Add, Zin
- Zout, R1in, End
-

Solution 2



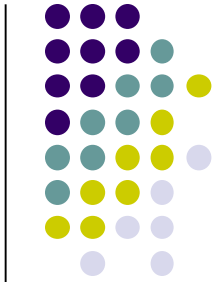
- PC out, MAR in, Read, Select4, Add , Zin
- Zout, PCin, Yin, WMFC
- MDRout, IR in,
- IRNUM out, MAR in, Read
- R1out, Yin, WMFC
- MDRout, Select Y, Add, Zin
- Zout, R1in, End

Solution 3



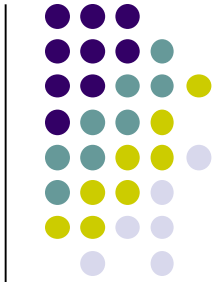
- PC out, MAR in, Read, Select4, Add , Zin
- Zout, PCin, Yin, WMFC
- MDRout, IR in
- IRNUM out, MAR in, Read
- WMFC
- MDRout, MAR in, Read
- R1out, Yin, WMFC
- MDRout , SelectY, Add, Zin
- Zout, R1in, End

Execution of Branch Instructions



- A branch instruction replaces the contents of PC with the branch target address, which is usually obtained by adding an offset X given in the branch instruction.
- The offset X is usually the difference between the branch target address and the address immediately following the branch instruction.
- UnConditional branch

Execution of Branch Instructions



Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	Offset-field-of-IR_{out}, Select Y, Add, Z_{in}
5	Z_{out}, PC_{in}, End

Figure 7.7. Control sequence for an unconditional branch instruction.